IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of:		
Sunil Kunisetty, et al.	)	Confirmation No.: 2092
Serial No.: 10/051,274	)	Examiner: Chrystine Pham
Filed on: January 22, 2002	)	Group Art Unit No: 7586

For: METHOD AND SOFTWARE FOR PROCESSING SERVER PAGES

MS Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Sir:

This Appeal Brief is submitted in support of the Notice of Appeal filed on April 16, 2007. A Pre-Appeal Brief Request for Review was filed with the Notice of Appeal. The period in which this Appeal Brief may be filed extends to June 16, 2007.

I. REAL PARTY IN INTEREST

Oracle International Corporation is the real party in interest.

II. RELATED APPEALS AND INTERFERENCES

The present application is not related to any other cases before the Board of Patent Appeals and Interferences.

III. STATUS OF CLAIMS

Claims 1, 2, 5, 6, 9, 11, 15-17, 19-21, and 23-34 have been finally rejected and are the subjects of this appeal. Claims 3-4, 7-8, 10, 11-14, and 22 have been canceled in response to previous actions.

IV. STATUS OF AMENDMENTS

Claims 1, 2, 5, 6, 9, 11, 15-17, 19-21, and 23-34 have not been amended after the Final Office Action.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present application contains independent Claims 1, 5, 15, and 19, which are summarized below. The claims summarized below are annotated to cross-reference features of the claims to specific examples of those features disclosed in the specification. However, the annotations are not intended to limit the scope of the recited features to those specific examples to which the annotations refer. Moreover, the annotations are not meant to be exhaustive.

Claim 1 recites (with added reference annotations in parenthesis) a computer-implemented method (paragraph [12]) of dynamically generating web pages (paragraphs [23]-[24]), said method comprising:

analyzing a page (Contents 133 in Fig. 1, paragraph [23]) that includes markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]) and a set of code instructions (paragraph [23]) executable on a server (paragraphs [22]-[23]);

extracting the markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]) from the page (Contents 133 in Fig. 1, paragraph [23]);

generating a servlet class (paragraph [23], [24], [25]) for the page based on the set of code instructions (paragraph [23], [24]), wherein the servlet class does not include the markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]-[25]);

loading a copy of the markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]) into shared memory (paragraphs [23], [26]);

in response to each request (paragraphs [22], [23], [24], [29]) of a plurality of requests (paragraphs [22]-[24], [29]) for the page (Contents 133 in Fig. 1, paragraph [23]) from a plurality of clients (Clients 101, 102, and 103 in FIG. 1, paragraphs [22]-[24], [29]), performing the steps of

instantiating a distinct instance of the servlet class (Instances 121, 123, and 125 in Fig. 1, paragraph [26]) on the server (Database 100 in Fig. 1, paragraphs [22]-[25]), wherein instantiating each instance (Instance 121, 123, and 125 in Fig. 1, paragraphs [24]-[25]) of the servlet class does not create another copy of the markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]);

executing (Instances 121, 123, and 125 in Fig. 1, paragraphs [25]-[26]) said distinct instance of the servlet class (Instances 121, 123, and 125 in Fig. 1), wherein execution of each instance of the server class generates (paragraphs [25]-[26]) a compiled page (paragraph [27]) based on the copy of the markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]) that resides in shared memory (paragraphs [23], [26]), and the set of code instructions (paragraph [23]); and

sending the compiled page to a client that requested the page (Clients 101, 102, and 103 in FIG. 1, paragraphs [22], [24], [29]).

Claim 5 recites a method (paragraph [12]) of initiating a first instance (Instances 121, 123, and 125 in Fig. 1, paragraph [26]) of an application (paragraphs [25]-[26]) that shares a set of markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]) with other instances of the application (Instances 121, 123, and 125 in Fig. 1, paragraph [26]), wherein the first instance of the application is generated by compiling code from a page (pargraphs [25]-[26]) that contains both the code (HotLoaded Class 141 in Fig. 1, paragraphs [22]-[24]) and the set of markup (HotLoaded Class 141 in Fig. 1, paragraph [23]) in response to a request (paragraphs [22], [23], [24], [29]) from one or more users (Clients 101, 102, and 103 in FIG. 1, paragraphs [22]-[24], [29]), said method comprising:

executing instructions to instantiate the first instance of the application (paragraphs [25]-[27]), wherein said instructions are stored on a computer-readable medium (paragraphs [41], [43]), said instructions that, when executed, cause one or more processors (paragraphs [41], 43]) to perform the steps of:

analyzing the page (Contents 133 in Fig. 1, paragraph [23]) to extract the set of markup text (HotLoaded Class 141 in Fig. 1, paragraph [23], [25]-[27], [35]-[40]);

generating a servlet class (paragraph [23], [24], [25]) for the page (Contents 133 in Fig. 1, paragraph [23]) based on the code (paragraphs [22]-[24]) from the page (paragraphs [25]-[26]), wherein the servlet class (paragraphs [23]-[25]) does not include the markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]-[25]);

loading a copy of the markup text (HotLoaded Class 141 in Fig. 1, paragraphs [23], [34]-[40]) into shared, read-only memory (paragraphs [23], [26]);

in response to each request (paragraphs [22], [23], [24], [29]) of a plurality of requests (paragraphs [22]-[24], [29]) for the page (Contents 133 in Fig. 1, paragraph [23]) from one or more users (Clients 101, 102, and 103 in FIG. 1, paragraphs [22]-[24], [29]), performing the steps of

instantiating a distinct instance of the servlet class (Instances 121, 123, and 125 in Fig. 1, paragraph [26]), wherein instantiating each instance of the servlet class does not create another copy of the markup text (HotLoaded Class 141 in Fig. 1, paragraphs [23], [34]-[40]);

executing said distinct instance (Instances 121, 123, and 125 in Fig. 1, paragraphs [25]-[26]) of the servlet class, wherein execution of each instance of the server class generates a compiled page paragraph [27]) based on the copy of the markup text (HotLoaded Class 141 in Fig. 1, paragraph [23], [34]-[40]) that resides in shared, read-only memory, and the set of code instructions;

sending the compiled page to a client that requested the page (Clients 101, 102, and 103 in FIG. 1, paragraphs [22], [24], [29]); and

accessing the set of markup text in the shared, read-only memory when the code from the first instance of the application is executed (paragraphs [34]-[40]).

Claim 15 recites a computer-readable storage medium bearing instructions that, when executed, cause one or more processors (paragraphs [41], [43]) to perform a method for sharing markup text from a page among a plurality of users in response to requests from said plurality of users (paragraphs [23]-[25]), said method comprising:

analyzing a page (Contents 133 in Fig. 1, paragraph [23]) that includes markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]) and a set of code instructions (paragraph [23]) executable on a server (paragraphs [22]-[23]);

extracting the markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]) from the page (Contents 133 in Fig. 1, paragraph [23]);

generating a servlet class (paragraphs [23], [24], [25]) for the page based on the set of code instructions (paragraphs [23], [24]), wherein the servlet class does not include the markup text (HotLoaded Class 141 in Fig. 1, paragraphs [23]-[25]);

loading a copy of the markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]) into shared memory (paragraphs [23], [26]);

in response to each request (paragraphs [22], [23], [24], [29]) of a plurality of requests (paragraphs [22]-[24], [29]) for the page (Contents 133 in Fig. 1, paragraph [23]) from a plurality of clients (Clients 101, 102, and 103 in FIG. 1, paragraphs [22]-[24], [29]), performing the steps of

instantiating a distinct instance of the servlet class (Instances 121, 123, and 125 in Fig. 1, paragraph [26]) on the server (Database 100 in Fig. 1, paragraphs [22]-[25]), wherein instantiating each instance (Instance 121, 123, and 125 in Fig. 1, paragraphs [24]-[25]) of the servlet class does not create another

copy of the markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]);

executing (Instances 121, 123, and 125 in Fig. 1, paragraphs [25]-[26]) said distinct instance of the servlet class (Instances 121, 123, and 125 in Fig. 1), wherein execution of each instance of the server class generates (paragraphs [25]-[26]) a compiled page (paragraph [27]) based on the copy of the markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]) that resides in shared memory (paragraphs [23], [26]), and the set of code instructions (paragraph [23]); and

sending the compiled page to a client that requested the page (Clients 101, 102, and 103 in FIG. 1, paragraphs [22], [24], [29])

Claim 19 recites a computer-readable storage medium bearing instructions that, when executed, cause one or more processors (paragraphs [41], [43]) to perform a method of initiating a first instance of an application that shares a set of markup text with other instances of the application, wherein the first instance of the application is generated by compiling code from a page that contains both the code and the set of markup text in response to a request from one or more users (paragraphs [23]-[25]), said method comprising:

executing instructions to instantiate the first instance of the application (paragraphs [25]-[27]);

wherein the instructions (paragraphs [41], [43]) to instantiate the first instance of the application include:

analyzing the page (Contents 133 in Fig. 1, paragraph [23]) to extract the set of markup text (HotLoaded Class 141 in Fig. 1, paragraph [23], [25]-[27], [35]-[40]);

generating a servlet class (paragraph [23], [24], [25]) for the page (Contents 133 in Fig. 1, paragraph [23]) based on the code (paragraphs [22]-[24]) from the page (paragraphs [25]-[26]), wherein the servlet class (paragraphs [23]-[25]) does not include the markup text (HotLoaded Class 141 in Fig. 1, paragraph [23]-[25]);

loading a copy of the markup text (HotLoaded Class 141 in Fig. 1, paragraphs [23], [34]-[40]) into shared, read-only memory (paragraphs [23], [26]);

in response to each request (paragraphs [22], [23], [24], [29]) of a plurality of requests (paragraphs [22]-[24], [29]) for the page (Contents 133 in Fig. 1, paragraph [23]) from one or more users (Clients 101, 102, and 103 in FIG. 1, paragraphs [22]-[24], [29]), performing the steps of

instantiating a distinct instance of the servlet class (Instances 121, 123, and 125 in Fig. 1, paragraph [26]), wherein instantiating each instance of the servlet class does not create another copy of the markup text (HotLoaded Class 141 in Fig. 1, paragraphs [23], [34]-[40]);

executing said distinct instance (Instances 121, 123, and 125 in Fig. 1, paragraphs [25]-[26]) of the servlet class, wherein execution of each instance of the server class generates a compiled page paragraph [27]) based on the copy of the markup text (HotLoaded

Class 141 in Fig. 1, paragraph [23], [34]-[40]) that resides in shared, read-only memory, and the set of code instructions;

sending the compiled page to a client that requested the page (Clients 101, 102, and 103 in FIG. 1, paragraphs [22], [24], [29]); and

accessing the set of markup text in the shared, read-only memory when the code from the first instance of the application is executed (paragraphs [34]-[40]).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-2, 5-6, 9, 11, 15-17, 19-21, and 23-34 stand rejected under 35 U.S.C. § 103(a) as being obvious, allegedly, under U.S. Patent Application 2002/0004813 A1 ("Agrawal") in view of U.S. Patent No. 6,675,354 ("Claussen").

VIII. ARGUMENTS

The Features of Claims 1-2, 5-6, 9, 11, 15-17, 19-21, and 23-34 are Not Disclosed, Taught, or Suggested by the Alleged Combination of Agrawal or Claussen

As will be seen from the discussion below, the Examiner's rejection of claims 1-2, 5, 6, 9, 11, 15-17, 19-21, and 23-34 under 35 U.S.C. § 103(a) as being allegedly unpatentable over Agrawal in view of Claussen do not establish a *prima facie* case of obviousness. There are clear errors of fact and of law in the Examiner's rejections that make the rejection improper and without basis.

INDEPENDENT CLAIM 1

Among other features, Claim 1 recites "instantiating a distinct instance of the servlet class on the server, wherein instantiating each instance of the servlet class does not create another

copy of the markup text". Neither Agrawal nor Claussen discloses, teaches, or suggests anything about "instantiating" a servlet class without creating "another copy of the markup text". This notion is not found anywhere in the cited references.

The Examiner relies on S62, S59, and S60 (and the associated text) in FIG. 5 to allegedly show that Agrawal discloses "instantiating a distinct instance of the servlet class on the server, wherein instantiating each instance of the servlet class does not create another copy of the markup text". However, these portions of Agrawal actually refer to retrieving cached blocks of a web page, not to static elements within a servlet class. As will be seen from the discussion below, there is absolutely no reason to assume that the cached blocks of Agrawal must correspond to an "instance of the servlet class [that] does not create another copy of the markup text".

Agrawal describes a system that analyzes web pages in order to find cacheable blocks within web pages. "For example, [in Agrawal], each of the blocks A, B, C, and D may be generated by code present in the underlying script of the document . . . The code that generates . . . A is necessarily distinct from the code in the script that generates. . . . B." (See Agrawal, paragraph [0033]). Essentially, Agrawal describes identifying blocks of code in an HTML web page, caching those blocks of code independently of a page, and retrieving those blocks of code when a page that contains the block is accessed. Thus, upon access of a web page that references cached code, the Agrawal system retrieves the cached block of code and uses that copy of HTML code to generate the web page. By doing so, the Agrawal system does not need to perform full web page caching. It can store a single block of code for elements on a web page that are repeated often (e.g., across multiple web pages). This is different from "instantiating a

distinct instance of the servlet class on the server, wherein instantiating each instance of the servlet class does not create another copy of the markup text" as recited in Claim 1.

For example, a difference between Agrawal and the cited element from Claim 1 is that the text being loaded into shared memory in Claim 1 is static to a servlet class. In other words, the text being loaded into shared memory in Claim 1 refers to a new way of handling text that would normally be part of a servlet class, but is stored in shared memory, for amont . As mentioned throughout the Specification, an example of use for the claimed invention is to handle and share Java literal strings. (See Specification, paragraphs [10]-[12] and [22]-[29]). The instantiating step refers to placing text that typically would be a part of the each instance of a servlet classes into a common, shared location accessible to multiple instances of the same servlet class. Thus, here, instead of storing the text from the servlet as literal strings, the text is hotloaded into a shared location available to multiple instances of a servlet class. This is not taught be Agrawal. Each block of cached code in Agrawal refers to elements of a page and not to strings within a servlet class. Put another way, an Agrawal block of code does not access text shared by multiple instances of the same servlet class.

In addition, the Appellants note (and this was admitted by the Examiner) that Agrawal does not teach the use of dynamically generated servlets. Accordingly, there is no need for Agrawal to "instantiat[e] a distinct instance of the servlet class on the server, wherein instantiating each instance of the servlet class does not create another copy of the markup text" since it does not use dynamically generated servlet classes into it.

Like Agrawal, Claussen does not teach "instantiating a distinct instance of the servlet class on the server, wherein instantiating each instance of the servlet class does not create another copy of the markup text" as recited in Claim 1. The Examiner states that "Claussen

discloses a system and method for serving dynamically generated servlet [class] using JSP to requesting clients." The Appellants respectfully submit that the Claussen reference describes the typical mechanisms used to generate servlet classes. In addition, it describes a document object model tree that identifies custom tags that aid users in debugging code. It does not, however, describe caching portions of text from a servlet classes so that multiple instances of the same class can access the same shared information, thereby, reducing the footprint of each servlet class.

Moreover, even if an alleged Agrawal-Claussen combination did teach all of the elements of Claim 1. There is no motivation to combine the two references. The Examiner states that "Agrawal and Claussen are analogous art because they are both directed to dynamically generating scripted pages. It would have been obvious to one of ordinary skill in the pertinent art at the time the invention was made to incorporate the teaching of Claussen into that of Agrawal for the inclusion of a servlet class. And the motivation for doing so would have been to allow error-handling code to be included in the servlet class for improved debugging and maintenance of the requested web pages." This motivation to combine does not make sense in light of the nature of Agrawal. Agrawal was designed to improve web server performance and limit the amount of processing the web server has to do. Adding additional debugging script to Agrawal would defeat many of the performance enhancements Agrawal is designed to accomplish. For at least these reasons, Claim 1 is patentable over Agrawal and Claussen, individually and in any alleged combination.

By virtue of their dependence from Claim 1, Claims 2 and 9 include the features of Claim 1 that are distinguished from Agrawal and Claussen above. As a result, Claims 2 and 9 are

patentable over an alleged Agrawal-Claussen combination under 35 U.S.C. § 103(a) for at least the reasons discussed above in connection with Claim 1.

DEPENDENT CLAIMS 23-25

Claims 23-25 recite additional features of the servlet class described in Claim 1. For example, Claim 23 recites that a servlet class includes an inner class. This is a feature clearly not taught by Agrawal. In fact, the Examiner does not even allege that Agrawal teaches an inner class. Instead, the Examiner argues that Claussen does at Col. 12:55-13:10. The Appellants point out that the cited portions of Claussen refer to techniques for using multiple scripting languages to generate pages and it does not describe generating an inner class for a servlet class as recited in Claim 23. Thus, the rejection of Claim 23 as being unpatentable in view of an alleged Agrawal and Claussen combination should be withdrawn.

Additionally, Claim 24 recites that loading markup text includes hot-loading an instance of the inner class. The text recited in Claim 1 is loaded into the inner class, which in turn is cached until a servlet needs access to hot-loaded text. This is also not described in either Claussen or Agrawal. The Examiner argues that Agrawal teaches this feature by citing various portions of Agrawal (e.g., paragraphs [0033], [0034], [0053], [0061], and [0064]) that describe blocks of code, however, none of the cited portions from Agrawal actually teach, or suggest an inner class (or equivalent thereof). Thus, the rejection of Claim 23 as being unpatentable in view of an alleged Agrawal and Claussen combination should be withdrawn.

Claim 25 recites that "the inner class includes an array of characters." The Examiner relies on Agrawal, paragraph [0061] to allegedly show that Agrawal teaches or suggests the inner class includes an array of characters. That portion of Agrawal makes absolutely no mention of

an inner class, let alone an inner class that includes an array of characters. Thus, the rejection of Claim 23 as being unpatentable in view of an alleged Agrawal and Claussen combination should be withdrawn.

For at least the reasons mentioned above, the Appellants respectfully request that the rejection of Claims 1, 2, 9, and 23-25 under 35 U.S.C. § 103(a) be withdrawn.

CLAIMS 15-17 AND 29-31

Claims 15-17 and 29-31 are computer-readable storage medium claims which contain limitations reasonably analogous to those described above. The Applicants submit that these claims are patentable over the alleged Agrawal-Claussen combination for at least the same reasons as given above.

INDEPENDENT CLAIM 5

Claim 5 recites, among other features, "instantiating a distinct instance of the servlet class, wherein instantiating each instance of the servlet class does not create another copy of the markup text". This element is reasonably analogous to the element in Claim 1 discussed above, and can be distinguished from Agrawal and Claussen for the same reasons. In addition, Claim 5 further recites "accessing the set of markup text in the shared, read-only memory when the code from the first instance of the application is executed."

Neither Agrawal nor Claussen discloses, teaches, or suggests anything about "accessing . . . text . . . when the code from the first instance of the application is executed." In fact, the Examiner does not discuss this element in the rejection of the claims. Accordingly, the Appellants cannot speculate as to what portions of Agrawal or Claussen the Examiner may or

may not have equated with this element. The Appellants do note that according to their understanding of Agrawal and Claussen, neither one discusses accessing the same set of text of a servlet class by separate instances of an application. For at least these reasons, Claim 5 is patentable over Agrawal and Claussen, individually and in any alleged combination.

By virtue of their dependence from Claim 5, Claims 6, 11, and 26-28 include the features of Claim 5 that are distinguished from Agrawal and Claussen above. Moreover, Claims 6, 11, 26-28 recite features reasonably analogous to those in Claims 2, 9, and 23-25 respectively. As a result, Claims 6, 11, and 26-28 are patentable over an alleged Agrawal-Claussen combination under 35 U.S.C. § 103(a) for at least the reasons discussed above.

CLAIMS 19-21 AND 32-34

Claims 19-21 and 32-34 are computer-readable storage medium claims which contain limitations reasonably analogous to those described above. The Applicants submit that these claims are patentable over the alleged Agrawal-Claussen combination for at least the same reasons as given above.

IX. CONCLUSION AND PRAYER FOR RELIEF

Based on the foregoing, it is respectfully submitted that the rejections of Claims 1, 2, 5, 6, 9, 11, 15-17, 19-21, and 23-34 lack the requisite factual and legal bases. Appellants respectfully request that the Honorable Board **reverse** the rejections of Claims 1, 2, 5, 6, 9, 11, 15-17, 19-21, and 23-34.

Respectfully submitted,

HICKMAN PALERMO TRUONG & BECKER LLP

Date: June 18, 2007

Joseph M. Olsen
Registration No. 58,724

2055 Gateway Place, Suite 550
San Jose, California 95110-1089
Tel: (408) 414-1224
Fax: (408) 414-1076

CLAIMS APPENDIX

1. A computer-implemented method of dynamically generating web pages, said method comprising:

analyzing a page that includes markup text and a set of code instructions executable on a server;

extracting the markup text from the page;

generating a servlet class for the page based on the set of code instructions, wherein the servlet class does not include the markup text;

loading a copy of the markup text into shared memory;

in response to each request of a plurality of requests for the page from a plurality of clients, performing the steps of

instantiating a distinct instance of the servlet class on the server, wherein instantiating each instance of the servlet class does not create another copy of the markup text;

executing said distinct instance of the servlet class, wherein execution of each instance of the server class generates a compiled page based on the copy of the markup text that resides in shared memory, and the set of code instructions; and

sending the compiled page to a client that requested the page.

2. The computer-implemented method according to claim 1, further comprising storing the markup text in a resource file associated with the application.

5. A method of initiating a first instance of an application that shares a set of markup text with other instances of the application, wherein the first instance of the application is generated by compiling code from a page that contains both the code and the set of markup in response to a request from one or more users, said method comprising:

executing instructions to instantiate the first instance of the application, wherein said instructions are stored on a computer-readable medium, said instructions that, when executed, cause one or more processors to perform the steps of:

analyzing the page to extract the set of markup text;

generating a servlet class for the page based on the code from the page, wherein the servlet class does not include the markup text;

loading a copy of the markup text into shared, read-only memory;

in response to each request of a plurality of requests for the page from the one or more users, performing the steps of

instantiating a distinct instance of the servlet class, wherein instantiating each instance of the servlet class does not create another copy of the markup text;

executing said distinct instance of the servlet class, wherein execution of each instance of the server class generates a compiled page based on the copy of the markup text that resides in shared, read-only memory, and the set of code instructions;

sending the compiled page to a client that requested the page; and

accessing the set of markup text in the shared, read-only memory when the code from the first instance of the application is executed.

6. A method according to claim 5, wherein the class is not loaded into the shared, read-only memory when the other instances of application are executed.

9. A computer-implemented method according to claim 1, wherein:
the markup text includes information to be displayed to a user and an annotation directing a user agent how to render the information to be displayed to the user; and
the markup output by the executing servlet class includes the annotation.

11. A method according to claim 5, wherein:
the set of markup text includes information to be displayed to a user and an annotation directing a user agent how to render the information to be displayed to the user; and
the set of markup output by the application includes the annotation.

15. A computer-readable storage medium bearing instructions that, when executed, cause one or more processors to perform a method for sharing markup text from a page among a plurality of users in response to requests from said plurality of users, said method comprising:
analyzing a page that includes markup text and a set of code instructions executable on a server;
extracting the markup text from the page;
generating a servlet class for the page based on the set of code instructions, wherein the servlet class does not include the markup text;

loading a copy of the markup text into shared memory;

in response to each request of a plurality of requests for the page from a plurality of clients, performing the steps of

instantiating a distinct instance of the servlet class on the server, wherein instantiating each instance of the servlet class does not create another copy of the markup text;

executing said distinct instance of the servlet class, wherein execution of each instance of the server class generates a compiled page based on the copy of the markup text that resides in shared memory, and the set of code instructions; and

sending the compiled page to a client that requested the page.

16. The computer-readable storage medium of claim 15, further comprising instructions to store the markup text in a resource file associated with the application.

17. The computer-readable storage medium of claim 15, wherein:

the markup text includes information to be displayed to a user and an annotation directing a user agent how to render the information to be displayed to the user; and

the markup output by the executing servlet class includes the annotation.

19. A computer-readable storage medium bearing instructions that, when executed, cause one or more processors to perform a method of initiating a first instance of an application that shares a set of markup text with other instances of the application, wherein the first instance of the application is generated by compiling code from a

page that contains both the code and the set of markup text in response to a request from one or more users, said method comprising:

executing instructions to instantiate the first instance of the application;

wherein the instructions to instantiate the first instance of the application include:

analyzing the page to extract the set of markup text;

generating a servlet class for the page based on the code from the page, wherein the servlet class does not include the markup text;

loading a copy of the markup text into shared, read-only memory;

in response to each request of a plurality of requests for the page from the one or more users, performing the steps of

instantiating a distinct instance of the servlet class, wherein instantiating each instance of the servlet class does not create another copy of the markup text;

executing said distinct instance of the servlet class, wherein execution of each instance of the server class generates a compiled page based on the copy of the markup text that resides in shared, read-only memory, and the set of code instructions;

sending the compiled page to the user that requested the page; and

accessing the set of markup text in the shared, read-only memory when the code from the first instance of the application is executed.

20. The computer-readable storage medium of claim 19, wherein the class is not loaded into the shared, read-only memory when the other instances of the application are executed.

21. The computer-readable storage medium of claim 19, wherein:

the set of markup text includes information to be displayed to a user and an annotation directing a user agent how to render the information to be displayed to the user; and

the set of markup output by the application includes the annotation.

23. A computer-implemented method according to claim 1, wherein the servlet class includes an inner class.

24. A computer-implemented method according to claim 23, wherein the step of loading a copy of the markup text includes hot-loading an instance of the inner class.

25. A computer-implemented method according to claim 24, wherein the inner class comprises an array of characters.

26. A method according to claim 5, wherein the servlet class includes an inner class.

27. A method according to claim 26 wherein the step of loading a copy of the markup text includes hot-loading an instance of the inner class.

28. A method according to claim 27, wherein the inner class comprises an array of characters.

29. A computer-readable storage medium according to claim 15, wherein the servlet class includes an inner class.

30. A computer-readable storage medium according to claim 29, wherein the step of loading a copy of the markup text includes hot-loading an instance of the inner class.

31. A computer-readable storage medium according to claim 30, wherein the inner class comprises an array of characters.

32. A computer-readable storage medium according to claim 19, wherein the servlet class includes an inner class.

33. A computer-readable storage medium according to claim 32, wherein the step of loading a copy of the markup text includes hot-loading an instance of the inner class.

34. A computer-readable storage medium according to claim 33, wherein the inner class comprises an array of characters.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.